World-set Decompositions: Expressiveness and Efficient Algorithms*

Lyublena Antova, Christoph Koch, and Dan Olteanu

Lehrstuhl für Informationssysteme
Universität des Saarlandes, Saarbrücken, Germany
{lublena, koch, olteanu}@infosys.uni-sb.de

Abstract. Uncertain information is commonplace in real-world data management scenarios. The ability to represent large sets of possible instances (worlds) while supporting efficient storage and processing is an important challenge in this context. The recent formalism of *world-set decompositions (WSDs)* provides a space-efficient representation for uncertain data that also supports scalable processing. WSDs are *complete* for finite world-sets in that they can represent any finite set of possible worlds. For possibly infinite world-sets, we show that a natural generalization of WSDs precisely captures the expressive power of c-tables. We then show that several important problems are efficiently solvable on WSDs while they are NP-hard on c-tables. Finally, we give a polynomial-time algorithm for factorizing WSDs, i.e. an efficient algorithm for minimizing such representations.

1 Introduction

Recently there has been renewed interest in incomplete information databases. This is due to the many important applications that systems for representing incomplete information have, such as data cleaning, data integration, and scientific databases.

Strong representation systems [19, 3, 18] are formalisms for representing sets of possible worlds which are closed under query operations in a given query language. While there have been numerous other approaches to dealing with incomplete information, such as closing possible worlds semantics using certain answers [1, 7, 12], constraint or database repair [13, 10, 9], and probabilistic ranked retrieval [14, 4], strong representation systems form a compositional framework that is minimally intrusive by not requiring to lose information, even about the lack of information, present in an information system: Computing certain answers, for example, entails a loss of possible but uncertain information. Strong representation systems can be nicely combined with the other approaches. For example, data transformation queries and data cleaning steps effected within a strong representation systems framework can be followed by a query with ranked retrieval or certain answers semantics, closing the possible worlds semantics.

The so-called *c-tables* [19, 16, 17] are the prototypical strong representation system. However, c-tables are not well suited for representing large incomplete

* This article is an extended version of the paper [6] that has appeared in the Proceedings of the International Conference on Database Theory (ICDT) 2007.

databases in practice. Two recent works presented strong, indeed *complete*, representation systems for finite sets of possible worlds. The approach of the *Trio x-relations* [8] relies on a form of intensional information ("lineage") only in combination with which the formalism is strong. In [5] large sets of possible worlds are managed using *world-set decompositions (WSDs)*. The approach is based on relational product decomposition to permit space-efficient representation. [5] describes a prototype implementation and shows the efficiency and scalability of the formalism in terms of storage and query evaluation in a large census data scenario with up to 2^{10^6} worlds, where each world stored is several GB in size.

Examples of world-set decompositions. As WSDs play a central role in this work, we next exemplify them using two manually completed forms that may originate from a census and which allow for more than one interpretation (Figure 1). For simplicity we assume that social security numbers consist of only three digits. For instance, Smith's social security number can be read either as "185" or as "785". We can represent the available information using a relation in which possible alternative values are represented in set notation (so-called or-sets):

(TID)	S	N	M
t_1	{ 185, 785 }	Smith	{ 1, 2 }
t_2	{ 185, 186 }	Brown	{ 1, 2, 3, 4 }

This or-set relation represents $2 \cdot 2 \cdot 2 \cdot 4 = 32$ possible worlds.

We now enforce the integrity constraint that all social security numbers be unique. For our example database, this constraint excludes 8 of the 32 worlds, namely those in which both tuples have the value 185 as social security number. This constraint excludes the worlds in which both tuples have the value 185 as social security number. It is impossible to represent the remaining 24 worlds using or-set relations. What we could do is store each world explicitly using a table called a *world-set relation* of a given set of worlds. Each tuple in this table represents one world and is the concatenation of all tuples in that world (see Figure 1).

A world-set decomposition is a decomposition of a world-set relation into several relations such that their product (using the product operation of relational algebra) is again the world-set relation. The world-set represented by our initial or-set relation can also be represented by the product

$t_1.S$
185
785

$\times$

$t_1.N$
Smith

$\times$

$t_1.M$
1
2

$\times$

$t_2.S$
185
186

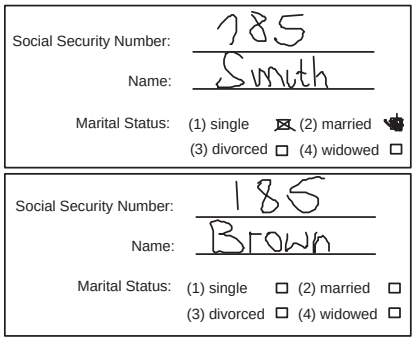
$\times$

$t_2.N$
Brown

$\times$

$t_2.M$
1
2
3
4

In the same way we can represent the result of data cleaning with the uniqueness constraint for the social security numbers as the product



$t_1.S$	$t_1.N$	$t_1.M$	$t_2.S$	$t_2.N$	$t_2.M$
185	Smith	1	186	Brown	1
185	Smith	1	186	Brown	2
185	Smith	1	186	Brown	3
185	Smith	1	186	Brown	4
185	Smith	2	186	Brown	1
		⋮			
785	Smith	2	186	Brown	4

Fig. 1. Two completed survey forms and a world-set relation representing the possible worlds with unique social security numbers.

$t_1.S$	$t_2.S$
185	186
785	185
785	186

$\times$

$t_1.N$
Smith

$\times$

$t_1.M$
1
2

$\times$

$t_2.N$
Brown

$\times$

$t_2.M$
1
2
3
4

One can observe that the result of this product is exactly the world-set relation in Figure 1. The decomposition is based on the *independence* between sets of fields, subsequently called *components*. Only fields that depend on each other, for example $t_1.S$ and $t_2.S$, belong to the same component. Since $\{t_1.S, t_2.S\}$ and $\{t_1.M\}$ are independent, they are put into different components.

WSDs can be naturally viewed as c-tables whose formulas have been put into a *normal form* represented by the component relations. The following c-table with global condition Φ is equivalent to the WSD with our integrity constraint enforced.

T	S	N	M	*cond*
				$\Phi = ((x = 185 \wedge z = 186) \vee (x = 785 \wedge z = 185) \vee$ $(x = 785 \wedge z = 186)) \wedge (y = 1 \vee y = 2) \wedge$ $(w = 1 \vee w = 2 \vee w = 3 \vee w = 4)$
	x	Smith	y	
	z	Brown	w	

Formal definitions of WSDs and c-tables will be given in the body of this article.

Contributions. The main goal of this work is to develop expressive yet efficient representation systems for infinite world-sets and to study the theoretical properties (such as expressive power, complexity of query-processing, and minimization) of these representation systems. Many of these results also apply to – and are new for – the world-set decompositions of [5].

Input	Representation system $\mathcal{W}$, instance $I = (R^I)$, tuple t	
Problems	Tuple Possibility:	$\exists \mathcal{A} \in \mathit{rep}(\mathcal{W}) : t \in R^{\mathcal{A}}$
	Tuple Certainty:	$\forall \mathcal{A} \in \mathit{rep}(\mathcal{W}) : t \in R^{\mathcal{A}}$
	Instance Possibility:	$\exists \mathcal{A} \in \mathit{rep}(\mathcal{W}) : R^I = R^{\mathcal{A}}$
	Instance Certainty:	$\forall \mathcal{A} \in \mathit{rep}(\mathcal{W}) : R^I = R^{\mathcal{A}}$
	Tuple Q-Possibility (query Q fixed):	$\exists \mathcal{A} \in \mathit{rep}(\mathcal{W}) : t \in Q(\mathcal{A})$
	Tuple Q-Certainty (query Q fixed):	$\forall \mathcal{A} \in \mathit{rep}(\mathcal{W}) : t \in Q(\mathcal{A})$
	Instance Q-Possibility (query Q fixed):	$\exists \mathcal{A} \in \mathit{rep}(\mathcal{W}) : R^I = Q(\mathcal{A})$
	Instance Q-Certainty (query Q fixed):	$\forall \mathcal{A} \in \mathit{rep}(\mathcal{W}) : R^I = Q(\mathcal{A})$

Table 1. Decision Problems for Representation Systems.

In [18], a strong argument is made supporting c-tables as a benchmark for the expressiveness of representation systems; we concur. Concerning efficient processing, we adopt v-tables as a lower bound regarding succinctness and complexity. The main development of this article is a representation system that combines, in a sense, the best of all worlds: (1) It is just as expressive as c-tables, (2) it is exponentially more succinct than unions of v-tables, and (3) on most classical decision problems, the complexity bounds are not worse than those for v-tables.

In more detail, the technical contributions of this article are as follows:

- We introduce gWSDs, an extension of the WSD model of [5] with variables and possibly negated equality conditions.
- We show that gWSDs are expressively equivalent to c-tables and are therefore a strong representation system for full relational algebra.
- We study the complexity of the main data management problems [3, 19] regarding WSDs and gWSDs, summarized in Table 1. Table 2 compares the complexities of these problems in our context to those of existing strong representation systems like the well-behaved ULDBs of Trio[1] and c-tables.
- We present an efficient algorithm for optimizing gWSDs, i.e., for computing an equivalent gWSD whose size is smaller than that of a given gWSD. In the case of WSDs, this is a minimization algorithm that produces the unique maximal decomposition of a given WSD.

One can argue that gWSDs are a practically more applicable representation formalism than c-tables: While having the same expressive power, many important problems are easier to solve. Indeed, as shown in Table 2, the complexity results for gWSDs on many important decision problems are identical to those for the much weaker v-tables. At the same time WSDs are still concise enough to support the space-efficient representation of very large sets of possible worlds (cf. the experimental evaluation on WSDs in [5]). Also, while gWSDs are strictly stronger than Trio representations, the complexity characteristics are better.

The results on finding maximal product decompositions relate to earlier work done by the database theory community on relational decomposition given

[1] The complexity results for Trio are from [8] and were not verified by the authors.

	v-tables [3]	gWSD	Trio [8]	c-tables [17]
Tuple Possibility	PTIME	PTIME	PTIME	NP-compl.
Tuple Certainty	PTIME	PTIME	PTIME	coNP-compl.
Instance Possibility	NP-compl.	NP-compl.	NP-hard	NP-compl.
Instance Certainty	PTIME	PTIME	NP-hard	coNP-compl.
Tuple Q-Possibility	NP-compl.	NP-compl.	?	NP-compl.
pos relational algebra	PTIME	PTIME	?	NP-compl.
Tuple Q-Certainty	coNP-compl.	coNP-compl.	?	coNP-compl.
pos relational algebra	PTIME	coNP-compl.	?	coNP-compl.
Instance Q-Possibility	NP-compl.	NP-compl.	NP-hard	NP-compl.
Instance Q-Certainty	coNP-compl.	coNP-compl.	NP-hard	coNP-compl.
pos relational algebra	PTIME	coNP-compl.	NP-hard	coNP-compl.

Table 2. Comparison of data complexities for standard decision problems.

schema constraints (cf. e.g. [2]). Our algorithms do not assume such constraints and only take a snapshot of a database at a particular point in time into consideration. Consequently, updates may require to alter a decomposition. Nevertheless, our results may be of interest independently from WSDs as for instance in certain scenarios with very dense relations, decompositions may be a practically relevant technique for efficiently storing and querying large databases.

Note that we do not consider probabilistic approaches to representing uncertain data (e.g. the recent work [14]) in this article. However, there is a natural and straightforward probabilistic extension of WSDs which directly inherits many of the properties studied in this article, see [5].

The structure of the article basically follows the list of contributions.

2 Preliminaries

We use the named perspective of the relational model and relational algebra with the operations selection σ, projection π, product $\times$, union $\cup$, difference $-$, and renaming δ.

A *relation schema* is a construct of the form $R[U]$, where R is a relation name and U is a nonempty set of attribute names.[2] Let $\mathbf{D}$ be an infinite set of atomic values, the *domain*. A *relation* over schema $R[A_1, \ldots, A_k]$ is a finite set of tuples $(A_1 : a_1, \ldots, A_k : a_k)$ where $a_1, \ldots, a_k \in \mathbf{D}$. A *relational schema* is a tuple $\Sigma = (R_1[U_1], \ldots, R_k[U_k])$ of relation schemas. A *relational structure (or database)* $\mathcal{A}$ over schema Σ is a tuple $(R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}})$, where each $R_i^{\mathcal{A}}$ is a relation over schema $R_i[U_i]$. When no confusion may occur, we will also use R rather than $R^{\mathcal{A}}$ to denote one particular relation over schema $R[U]$. For a relation R, $\mathsf{sch}(R)$ denotes the set of its attributes, $ar(R)$ its arity and $|R|$ the number of tuples in R.

A set of *possible worlds* (or *world-set*) over schema Σ is a set of databases over schema Σ. Let $\mathbf{W}$ be a set of finite structures, and let *rep* be a function that maps each $\mathcal{W} \in \mathbf{W}$ to a world-set of the same schema. Then $(\mathbf{W}, rep)$ is

[2] For technical reasons involving the WSDs presented later, we exclude nullary relations and will represent these (e.g., when obtained as results from a Boolean query) using unary relations over a special constant "true".

called a *strong representation system* for a query language if, for each query Q of that language and each $\mathcal{W} \in \mathbf{W}$ such that the schema of Q is consistent with the schema of the worlds in $rep(\mathcal{W})$, there is a structure $\mathcal{W}' \in \mathbf{W}$ such that $rep(\mathcal{W}') = \{Q(\mathcal{A}) \mid \mathcal{A} \in rep(\mathcal{W})\}$.

2.1 Tables

We now review a number of representation systems for incomplete information that are known from earlier work (cf. e.g. [17, 2]).

Let $\mathbf{X}$ be a set of variables. We call an equality of the form $x = c$ or $x = y$, where x and y are variables from $\mathbf{X}$ and c is from $\mathbf{D}$ an *atomic condition*, and will define *(general) conditions* as Boolean combinations (using conjunction, disjunction, and negation) of atomic conditions.

Definition 1 (c-table). A *c-multitable* [19, 17] over schema $(R_1[U_1], \ldots, R_k[U_k])$ is a tuple

$$\mathcal{T} = (R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$$

where each $R_i^{\mathcal{T}}$ is a set of $ar(R_i)$-tuples over $\mathbf{D} \cup \mathbf{X}$, $\phi^{\mathcal{T}}$ is a Boolean combination over equalities on $\mathbf{D} \cup \mathbf{X}$ called the *global condition*, and function $\lambda^{\mathcal{T}}$ assigns each tuple from one of the relations $R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}$ to a condition (called the *local condition* of the tuple). A c-multitable with $k = 1$ is called a *c-table*.

The semantics of a c-multitable $\mathcal{T}$, called its *representation* $rep(\mathcal{T})$, is defined via the notion of a valuation of the variables occurring in $\mathcal{T}$ (i.e., those in the tuples as well as those in the conditions). Let $\nu : \mathbf{X} \to \mathbf{D}$ be a valuation that assigns each variable in $\mathcal{T}$ to a domain value. We overload ν in the natural way to map tuples and conditions over $\mathbf{D} \cup \mathbf{X}$ to tuples and formulas over $\mathbf{D}$.[3] A *satisfaction* of $\mathcal{T}$ is a valuation ν such that $\nu(\phi^{\mathcal{T}})$ is true. A satisfaction ν takes $\mathcal{T}$ to a relational structure $\nu(\mathcal{T}) = (R_1^{\nu(\mathcal{T})}, \ldots, R_k^{\nu(\mathcal{T})})$ where each relation $R_i^{\nu(\mathcal{T})}$ is obtained as $R_i^{\nu(\mathcal{T})} := \{\nu(t) \mid t \in R_i^{\mathcal{T}} \wedge \nu(\lambda^{\mathcal{T}}(t)) \text{ is true}\}$. The representation of $\mathcal{T}$ is now given by its satisfactions, $rep(\mathcal{T}) := \{\nu(\mathcal{T}) \mid \nu \text{ is a satisfaction of } \mathcal{T}\}$. □

Proposition 1 ([19]). *The c-multitables are a strong representation system for relational algebra.*

We consider two important restrictions of c-multitables.

1. By a *g-multitable* [3], we refer to a c-multitable in which the global condition $\phi^{\mathcal{T}}$ is a conjunction of possibly negated equalities and $\lambda^{\mathcal{T}}$ maps each tuple to "true".
2. A *v-multitable* is a g-multitable in which the global condition $\phi^{\mathcal{T}}$ is a conjunction of equalities.

[3] Done by extending ν to be the identity on domain values and to commute with the tuple constructor, the Boolean operations, and equality.

Without loss of generality, we may assume that the global condition of a g-multitable is a conjunction of *negated qualities* and the global condition of a v-multitable is simply "true".[4] Subsequently, we will always assume these two normal forms and omit local conditions from g-multitables and both global and local conditions from v-multitables.

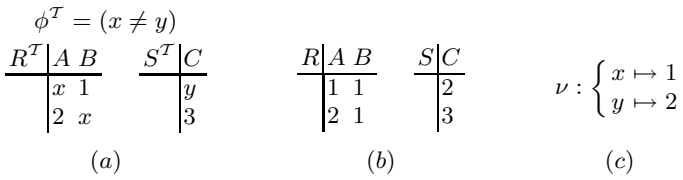


Fig. 2. A g-multitable $\mathcal{T}$ (a), possible world $\mathcal{A}$ (b), and a valuation s.t. $\nu(\mathcal{T}) = \mathcal{A}$ (c).

Example 1. Consider the g-multitable $\mathcal{T} = (R^{\mathcal{T}}, S^{\mathcal{T}}, \phi^{\mathcal{T}})$ of Figure 2 (a). Then the valuation of Figure 2 (c) satisfies the global condition of $\mathcal{T}$, as $\nu(x) \neq \nu(y)$. Thus $\mathcal{A} \in rep(\mathcal{T})$, where $\mathcal{A}$ is the structure from Figure 2 (b). □

Remark 1. It is known from [19] that v-tables are not a strong representation system for relational selection, but for the fragment of relational algebra built from projection, product, and union.

The definition of c-multitables used here is from [17]. The original definition from [19] has been more restrictive in requiring the global condition to be "true". While c-tables without a global condition are strictly weaker (they cannot represent the empty world-set), they nevertheless form a strong representation system for relational algebra.

In [2], the global conditions of c-multitables are required to be conjunctions of possibly negated equalities. It will be a corollary of a result of this paper (Theorem 2) that this definition is equivalent to c-multitables with arbitrary global conditions. □

We next define a restricted form of c-tables, called *mutex-tables*, or x-tables for short. This formalism is of particular importance in this paper as it is closely related to gWSDs, our main representation formalism. An x-table is a c-table where the global condition is a conjunction of negated equalities and the local conditions are conjunctions of equalities and a special form of negated equalities. We make this more precise next.

Consider a set of variables $\mathbf{Y}$ and a function $\mu : \mathbf{Y} \mapsto \mathbb{N}^+$ mapping variables to positive numbers. The *mutex set* $\mathbb{M}(\mathbf{Y}, \mu)$ for $\mathbf{Y}$ and μ is defined by

$$\{true\} \cup \{(x = i) \mid x \in \mathbf{Y}, 1 \leq i \leq \mu(x)\} \cup \{(x \neq 1 \wedge \ldots \wedge x \neq \mu(x)) \mid x \in \mathbf{Y}\}.$$

[4] Each g-multitable resp. v-multitable can be reduced to one in this normal form by variable replacement and the removal of tautologies such as $x = x$ or $1 = 1$ from the global condition.

Intuitively, $\mathbb{M}$ defines for each variable of $\mathbf{Y}$ possibly negated equalities such that a variable valuation satisfies precisely one of these conditions.

Definition 2 (mutex-table). A *(mute)x-multitable* is a c-multitable

$$\mathcal{T} = (R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}}),$$

where (1) the global condition $\phi^{\mathcal{T}}$ is a conjunction of negated equalities, (2) all local conditions defined by $\lambda^{\mathcal{T}}$ are conjunctions over formulas from a mutex set $\mathbb{M}(\mathbf{Y}, \mu)$ and equalities over $\mathbf{X} \cup \mathbf{D}$, and (3) the variables $\mathbf{Y}$ do not occur in $R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}$. An x-multitable with $k = 1$ is called an *x-table*. □

It will be a corollary of joint results of this paper (Lemma 1 and Theorem 2) that x-multitables are as expressive as c-multitables.

Proposition 2. *The x-multitables capture the c-multitables.*

This result implies that x-tables are a strong representation system for relational algebra. In this paper, however, we will make particular use of a weaker form of strongness, namely for positive relational algebra, in conjunction with efficient query evaluation.

Proposition 3. *The x-multitables are a strong representation system for positive relational algebra. The evaluation of positive relational algebra queries on x-multitables is in PTIME.*

Proof. We use the algorithm of [19, 17] for the evaluation of relational algebra queries on c-multitables and obtain an answer c-multitable of polynomial size. Consider a positive relational algebra query Q and c-multitables $\mathcal{T}$ and $\mathcal{T}'$, where $\mathcal{T}'$ represents the answer to Q on $\mathcal{T}$. We compute $\mathcal{T}'$ by recursively applying each operator in Q. The evaluation follows the relational case except for the computation of global and local conditions (which do not exist in the relational case). The global condition of $\mathcal{T}$ becomes the global condition of $\mathcal{T}'$. For projection and union, tuples preserve their local conditions from the input. In case of selection, the local condition of a result tuple is the conjunction of the local condition of the input tuple and, if required by the selection condition, of new equalities involving variables in the tuple and constants from the positive selections of Q. In case of product, the local condition of a result tuple is the conjunction of the local conditions of the constituent input tuples.

The local conditions in $\mathcal{T}'$ are thus conjunctions of local conditions of $\mathcal{T}$ and possibly additional equalities. In case $\mathcal{T}$ is an x-table, then its local conditions are conjunctions over formulas from a mutex set $\mathbb{M}$ and further equalities. Thus the local conditions of $\mathcal{T}'$ are also conjunctions over formulas from $\mathbb{M}$ and further equalities. $\mathcal{T}'$ is then an x-table. □

3 Representation Systems

This section develops the notion of world-set decompositions from tables and studies some of their basic properties.

3.1 Tabsets and Tabset Tables

We consider finite sets of multitables as representation systems, and will refer to such constructs as *tabsets* (rather than as *multitable-sets*, to be short).

A c-(resp., g-, v-)tabset $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ is a finite set of c-(g-, v-)multitables. The representation of a tabset is the union of the representations of the constituent multitables,

$$rep(\mathbf{T}) := rep(\mathcal{T}_1) \cup \cdots \cup rep(\mathcal{T}_n).$$

We next construct an *inlined* representation of a tabset as a single table by turning each multitable into a single tuple.

Let $\mathbf{A}$ be a g-tabset over schema Σ. For each $R[U]$ in Σ, let $|R|_{\max} = \max\{|R^{\mathcal{A}}| : \mathcal{A} \in \mathbf{A}\}$ denote the maximum cardinality of R in any multitable of $\mathbf{A}$. Given a g-multitable $\mathcal{A} \in \mathbf{A}$ with $R^{\mathcal{A}} = \{t_1, \ldots, t_{|R^{\mathcal{A}}|}\}$, let $\text{inline}(R^{\mathcal{A}})$ be the tuple obtained as the concatenation (denoted $\circ$) of the tuples of $R^{\mathcal{A}}$ padded with a special symbol $\perp$ up to arity $|R|_{\max} \cdot ar(R)$,

$$\text{inline}(R^{\mathcal{A}}) := t_1 \circ \cdots \circ t_{|R^{\mathcal{A}}|} \circ (\underbrace{\perp, \ldots, \perp}_{(|R|_{\max} - |R^{\mathcal{A}}|) \cdot ar(R)}).$$

Then tuple

$$\text{inline}(\mathcal{A}) := \text{inline}(R_1^{\mathcal{A}}) \circ \cdots \circ \text{inline}(R_{|\Sigma|}^{\mathcal{A}})$$

encodes all the information in $\mathcal{A}$.

Definition 3 (gTST). The *g-tabset table (gTST)* of a g-tabset $\mathbf{A}$ is the pair (W, λ) consisting of the table[5] $W = \{\text{inline}(\mathcal{A}) \mid \mathcal{A} \in \mathbf{A}\}$ and the function λ which maps each tuple $\text{inline}(\mathcal{A})$ of W to the global condition of $\mathcal{A}$. □

A vTST (TST) is obtained in strict analogy, omitting λ (λ and variables).

To compute $\text{inline}(R^{\mathcal{A}})$, we have fixed an arbitrary order of the tuples in $R^{\mathcal{A}}$. We represent this order by using indices d_i to denote the i-th tuple in $R^{\mathcal{A}}$ for each g-multitable $\mathcal{A}$, if that tuple exists. Then the TST has schema

$$\{R.d_i.A_j \mid R[U] \text{ in } \Sigma, 1 \leq i \leq |R|_{\max}, A_j \in U\}.$$

Example 2. An example translation from a tabset to a TST is given in Figure 3.

The semantics of a gTST (W, λ) as a representation system is given in strict analogy with tabsets,

$$rep(W, \lambda) := \bigcup \{rep(\text{inline}^{-1}(t), \lambda(t)) \mid t \in W\}.$$

Remark 2. Computing the inverse of "inline" is an easy exercise. In particular, we map $\text{inline}(R^{\mathcal{A}})$ to $R^{\mathcal{A}}$ as

$$(a_1, \ldots, a_{ar(R) \cdot |R|_{\max}}) \mapsto \{(a_{ar(R) \cdot k+1}, \ldots, a_{ar(R) \cdot (k+1)}) \mid 0 \leq k < |R|_{\max}, \\ a_{ar(R) \cdot k+1} \neq \perp, \ldots, a_{ar(R) \cdot (k+1)} \neq \perp\}.$$

[5] Note that this table may contain variables and occurrences of the $\perp$ symbol.

$\phi^{\mathcal{A}}$				$\phi^{\mathcal{B}}$				$\phi^{\mathcal{C}}$			
$R^{\mathcal{A}}$	A	B	$S^{\mathcal{A}}$ C	$R^{\mathcal{B}}$	A	B	$S^{\mathcal{B}}$ C	$R^{\mathcal{C}}$	A	B	$S^{\mathcal{C}}$ C
	a_1	a_2	a_5		b_1	b_2			c_1	c_2	c_3
	a_3	a_4	a_6		b_3	b_4					c_4
					b_5	b_6					c_5

(a) Three $(R[A, B], S[C])$-multitables $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$.

$R.d_1.A$	$R.d_1.B$	$R.d_2.A$	$R.d_2.B$	$R.d_3.A$	$R.d_3.B$	$S.d_1.C$	$S.d_2.C$	$S.d_3.C$	λ
a_1	a_2	a_3	a_4	$\bot$	$\bot$	a_5	a_6	$\bot$	$\phi^{\mathcal{A}}$
b_1	b_2	b_3	b_4	b_5	b_6	$\bot$	$\bot$	$\bot$	$\phi^{\mathcal{B}}$
c_1	c_2	$\bot$	$\bot$	$\bot$	$\bot$	c_3	c_4	c_5	$\phi^{\mathcal{C}}$

(b): TST of tabset $\{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$.

Fig. 3. Translation from a tabset (a) to a TST (b).

By construction, the gTST capture the g-tabsets.

Proposition 4. *The g-(resp., v-)TST capture the g-(v-)tabsets.*

Finally, there is an noteworthy normal form for gTSTs.

Proposition 5. *The gTST in which λ maps each tuple to a common global condition ϕ unique across the gTST, that is, $\lambda : \cdot \mapsto \phi$, capture the gTST.*

Proof. Given a g-tabset $\mathbf{A}$, we may assume without loss of generality that no two g-multitables from $\mathbf{A}$ share a common variable, either in the tables or the conditions, and that all global conditions in $\mathbf{A}$ are satisfiable. (Otherwise we could safely remove some of the g-multitables in $\mathbf{A}$.) But, then, ϕ is simply the conjunction of the global conditions in $\mathbf{A}$. For any tuple t of the gTST of $\mathbf{A}$, the g-multitable $(\text{inline}^{-1}(t), \phi)$ is equivalent to $(\text{inline}^{-1}(t), \lambda(t))$. □

Proviso. We will in the following write gTSTs as pairs (W, ϕ), where W is the table and ϕ is a single global condition shared by the tuples of W.

3.2 World-set Decompositions

We are now ready to define world-set decompositions, our main vehicle for efficient yet expressive representation systems.

A *product m-decomposition* of a relation R is a set of non-nullary relations $\{C_1, \ldots, C_m\}$ such that $C_1 \times \cdots \times C_m = R$. The relations $C_1, \ldots, C_m$ are called *components*. A product m-decomposition of R is *maximal(ly decomposed)* if there is no product n-decomposition of R with $n > m$.

Definition 4 (attribute-level gWSD). Let (W, ϕ) be a gTST. Then an *attribute-level world-set m-decomposition (m-gWSD)* of (W, ϕ) is a pair of a product m-decomposition of W together with the global condition ϕ. □

R	A	B
d_1	1	2
d_2	5	6

R	A	B
d_1	1	2

R	A	B
d_1	3	4
d_2	5	6

R	A	B
d_1	3	4

C_1	$R.d_1.A$	$R.d_1.B$
	1	2
	3	4

$\times$

C_2	$R.d_2.A$	$R.d_2.B$
	5	6
	$\bot$	$\bot$

Fig. 4. Four worlds and a corresponding 2-WSD.

We also consider two important simplifications of (attribute-level) gWSDs, those without global condition (called vWSDs), and vWSDs without variables (called WSDs). An example of a WSD is shown in Figure 4.

The semantics of a gWSD is given by its exact correspondence with a gTST,

$$rep\underbrace{(\{C_1,\ldots,C_m\},\phi)}_{\text{gWSD}} := rep\underbrace{(C_1\times\cdots\times C_m,\phi)}_{\text{gTST}}.$$

To decompose W, we treat its variables and the $\bot$-value as constants. Clearly, **A** and any gWSD of **A** represent the same set of possible worlds.

It immediately follows from the definition of WSDs that

Proposition 6. *Any finite set of possible worlds can be represented as a* 1*-WSD.*

Corollary 1. *WSDs are a strong representation system for any relational query language.*

The lack of power to express negated equalities, despite the ability to express disjunction, keeps vWSDs (and thus equally v-tabsets) from being strong.

Proposition 7. *vWSDs are a strong representation system for projection, product and union but do not form a strong representation system for selection or difference.*

Proof. We show that v-tabsets are a strong representation system for projection, product and union but not for selection. From the equivalence of v-tabsets and vWSDs the property also holds for vWSDs.

Let $\mathcal{T}=\{\mathcal{T}_1,\ldots,\mathcal{T}_n\}$ be a v-tabset of multitables over schema $(R_1,\ldots,R_m)$. The results of the operations projection $\pi_U(R)$, product $R_1\times R_2$ and union $R_1\cup R_2$ on $\mathcal{T}$, respectively, are then defined as

- $\pi_U(R)(\mathcal{T})=\{R' \mid \mathcal{T}_i\in\mathcal{T}, R'=\pi_U(R^{\mathcal{T}_i})\}$
- $(R_1\cup R_2)(\mathcal{T})=\{R' \mid \mathcal{T}_i\in\mathcal{T}, R'=R_1^{\mathcal{T}_i}\cup R_2^{\mathcal{T}_i}\}$
- $(R_1\times R_2)(\mathcal{T})=\{R' \mid \mathcal{T}_i\in\mathcal{T}, R'=R_1^{\mathcal{T}_i}\times R_2^{\mathcal{T}_i}\}$

To show that v-tabsets do not support selection, let $\mathcal{T}$ be a v-tabset consisting of a single v-table R:

R	A	B
d_1	x	2
d_2	1	x

Consider the selection query $q_1 := \sigma_{A=1}(R)$ which we evaluate on $\mathcal{T}$. The resulting world-set consists of the world $\{\langle A:1, B:2\rangle, \langle A:1, B:1\rangle\}$ for the case that the value of x is 1 and the worlds $\{\langle A:1, B:c\rangle\}$, where the c are constant different from 1 for the cases where x takes values c other than 1. That is, in the worlds where $x \neq 1$, q_1 returns only the second tuple of R. Let us call this result world-set W.

We prove by contradiction that there is no v-tabset representing precisely the world-set W. Since W is an infinite set of worlds and a v-tabset consists of only finitely many v-tables, there must be at least one v-table T that represents infinitely many worlds of the form $\{\langle A:1, B:c\rangle \mid c \in D\}$ and $rep(T) \subseteq W$. Since all tuples in a world of W have 1 as a value for A, all tuples in T must have it too, otherwise T will represent worlds that are not in W. Also, to represent infinitely many worlds, T must contain at least one variable. Assume first T has a single tuple. Then it must be of the form $\langle A:1, B:x\rangle$, where x is a variable. But then T represents the world $\langle A:1, B:1\rangle$, which is not in W.

Then T must have at least two tuples and each tuple is of the form $\langle A:1, B:x\rangle$, where x is a variable or a constant. Let $\langle A:1, B:x\rangle$ and $\langle A:1, B:y\rangle$ be two tuples of T with x and y in $\mathbf{D} \cup \mathbf{X}$.

If at least one of x and y is a variable, for example x, then the worlds where $x = 3$ are not contained in W.

If both x and y are constants, then we either have $x = y = c$ and $c \neq 1$ (to represent the world with one tuple from W), or $x = 2$ and $x = 1$ (to represent the world with two tuples from W). In both cases, for T to represent infinitely many worlds from W, there must be a tuple $\langle 1, z\rangle$ in T where z is a variable. However, the worlds where $z = 4$ are not in W. Contradiction.

Let now $R' := R \backslash S$ be the result of the difference operation applied on the v-multitable (R, S), where R is the v-table from the previous example:

R	A	B
d_1	x	2
d_2	1	x

S	A	B
d_3	1	1

The resulting world-set W consists of worlds with different number of tuples for R' depending on the value of x. In the worlds where $x = 1$ R' contains the tuple $\langle A:1, B:2\rangle$, and in the worlds where $x \neq 1$ R' is $\{\langle A:c, B:2\rangle, \langle A:1, B:c\rangle\}$ with c a constant different from 1. Using similar arguments as in the selection case we show that there is no single v-table T that represents infinitely many worlds of the form $\{\langle A:c, B:2\rangle, \langle A:1, B:c\rangle \mid c \in D\}$ and $rep(T) \subseteq W$. □

Remark 3. Note that verifying nondeterministically that a structure $\mathcal{A}$ is a possible world of gWSD $(\{C_1, \ldots, C_m\}, \phi)$ is easy: all we need is choose one tuple from each of the component tables $C_1, \ldots, C_m$, concatenate them into a tuple t, and check whether a valuation exists that satisfies ϕ and takes $\text{inline}^{-1}(t)$ to $\mathcal{A}$.

Already the vWSDs are exponentially more succinct than the v-tabsets. As is easy to verify,

Proposition 8. *Any v-tabset representation of the WSD*

$$\left\{ \begin{array}{c|c} C_1 & R.d_1.A \\ \hline & a_1 \\ & b_1 \end{array} \quad \cdots \quad \begin{array}{c|c} C_n & R.d_n.A \\ \hline & a_n \\ & b_n \end{array} \right\}$$

where the a_i, b_i are distinct domain values takes space exponential in n.

This greater succinctness is obtained at a price:

Theorem 1. *Given an attribute-level (g)WSD $\mathcal{W}$, checking whether the empty world is in rep($\mathcal{W}$) is NP-complete.*

Proof. To prove this, we show that the problem is in NP for attribute-level gWSDs and NP-hard for attribute-level WSDs.

Let $\mathcal{W} = (\{C_1, \ldots, C_n\}, \phi)$ be a gWSD. The problem is in NP since we can nondeterministically check whether there is a choice of component tuples $t_1 \in C_1, \ldots, t_n \in C_n$ such that $t_1 \circ \cdots \circ t_n$ represents the empty world.

The proof of NP-hardness is by reduction from Exact Cover by 3-Sets (X3C) [15]. Given a finite set X of size $|X| = 3q$ and a set C of three-element subsets of X.

Construction. We construct an attribute-level WSD $\{C_1, \ldots, C_q\}$ as follows. Let C_i be a table of schema $C_i[t_1.A_i, \ldots, t_{|X|}.A_i]$ with tuples $(t_1.A_i : a_1, \ldots, t_{|X|}.A_i : a_{|X|})$ for each $S \in C$ such that $a_j = \bot$ if $j \in S$ and $a_j = 1$ otherwise.

Correctness. This is straightforward to show, but note that each tuple of a component relation contains exactly three $\bot$ symbols. The WSD represents a set of worlds in which each one contains, naively, up to $3 \cdot q$ tuples. The composition of q component tuples $w_1 \in C_1, \ldots, w_q \in C_q$ can only represent the empty world if the $\bot$ symbols in $w_1, \ldots, w_q$ do not overlap. This guarantees that $w_1 \circ \cdots \circ w_q$ represents the empty set *only if* the sets from C corresponding to $w_1, \ldots, w_q$ form an *exact* cover of X.

It follows that the problem of deciding whether the q-ary tuple $(1, \ldots, 1)$ or whether the world containing just that tuple is *uncertain* is NP-complete. □

Note that this NP-hardness is a direct consequence of the succinctness increase in gWSDs as compared to gTSTs. On gTSTs, checking for the empty world is a trivial operation.

Example 3. We give an example of the previous reduction from X3C to testing whether the empty world is in the representation of a WSD. Let $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and let $C = \{\{1, 5, 9\}, \{2, 5, 8\}, \{3, 4, 6\}, \{2, 7, 8\}, \{1, 6, 9\}\}$. Then the WSD $\{C_1, C_2, C_3\}$ with each C_i the table

C_i	$d_1.A_i$	$d_2.A_i$	$d_3.A_i$	$d_4.A_i$	$d_5.A_i$	$d_6.A_i$	$d_7.A_i$	$d_8.A_i$	$d_9.A_i$
	$\bot$	1	1	1	$\bot$	1	1	1	$\bot$
	1	$\bot$	1	1	$\bot$	1	1	$\bot$	1
	1	1	$\bot$	$\bot$	1	$\bot$	1	1	1
	1	$\bot$	1	1	1	1	$\bot$	$\bot$	1
	$\bot$	1	1	1	1	$\bot$	1	1	$\bot$

for $1 \leq i \leq 3$ represents the empty world. Therefore, the first, third and fourth sets in C are an exact cover of X. □

Corollary 2. *Tuple certainty is coNP-hard for attribute-level WSDs.*

This problem remains in coNP even for general gWSDs. Nevertheless, since computing certain answers is a central task related to incomplete information, we will consider also the following restriction of gWSDs. As we will see, this alternative definition yields a representation system in which the tuple and instance certainty problems are in polynomial time while the formalism is still exponentially more succinct than gTSTs.

Definition 5 (gWSD). An attribute-level gWSD is called a *tuple-level gWSD* if for any two attributes A_i, A_j from the schema of relation R, and any tuple id d, the attributes $R.d.A_i, R.d.A_j$ of the component tables are in the same component schema. □

In other words, in tuple-level gWSDs, values for one and the same tuple cannot be split across several components – that is, here the decomposition is less fine-grained than in attribute-level gWSDs. In the remainder of this article, we will exclusively study tuple-level (g-, resp. v-)WSDs, and will refer to them as just simply (g-, v-)WSDs. Obviously, tuple-level (g)WSDs are just as expressive as attribute-level (g)WSDs, since they all are just decompositions of 1-(g)WSDs.

However, tuple-level (g)WSDs are less succinct than attribute-level (g)WSDs. For example, any tuple-level WSD equivalent to the attribute-level WSD

$$\left\{ \begin{array}{c|c} C_1 & R.d.A_1 \\ \hline & a_1 \\ & b_1 \end{array} \quad \cdots \quad \begin{array}{c|c} C_n & R.d.A_n \\ \hline & a_n \\ & b_n \end{array} \right\}$$

must be exponentially larger. Note that the WSDs of Proposition 8 are tuple-level.

4 Main Expressiveness Result

In this section we study the expressive power of gWSDs. We show that gWSDs and c-multitables are equivalent in expressive power, that is, for each gWSD one can find an equivalent c-multitable that represents the same set of possible worlds and vice versa. Thus, gWSDs form a strong representation system for relational algebra.

Theorem 2. *The gWSDs capture the c-multitables.*

Corollary 3. *gWSDs are a strong representation system for relational algebra.*

Corollary 4. *The g-tabsets capture the c-tabsets.*

That is, disjunction on the level of entire tables plus conjunctions of negated equalities as global conditions, as present in g-tables, are enough to capture the full expressive power of c-tables. In particular, we are able to eliminate all local conditions.

We prove Theorem 2 by providing a translation of gWSDs into x-multitables, a syntactically restricted form of c-multitables, and a translation of c-multitables into gWSDs.

Lemma 1. *gWSDs can be translated in linear time into equivalent x-multitables.*

Proof. Let $\mathcal{W} = (\{C_1, \ldots, C_m\}, \phi)$ be a (tuple-level) m-gWSD over relational schema $(R_1[U_1], \ldots, R_k[U_k])$.

Construction. In case a component C_j is empty, then $\mathcal{W}$ represents the empty world-set. Then $\mathcal{W}$ is equivalent to any x-(multi)table with an unsatisfiable global condition, i.e., $x \neq x$.

We consider next the case when all components C_j are non-empty, i.e., $n_j > 0$ for all $1 \leq j \leq m$. Let $C_j = \{w_1, \ldots, w_{n_j}\}$. We define a translation f from $\mathcal{W}$ to an equivalent c-multitable $\mathcal{T} = (R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$ in the following way.

1. The global condition ϕ of $\mathcal{W}$ becomes the global condition $\phi^{\mathcal{T}}$ of the x-multitable $\mathcal{T}$.
2. For each relation schema $R[U]$ we create a table $R^{\mathcal{T}}$ with the same schema.
3. We translate each component $C_j = \{w_1, \ldots, w_{n_j}\}$ of $\mathcal{W}$ in the following way. Let $w_i \in C$. Let d be a tuple identifier for a relation R defined in C_j and t be the tuple for d in w_i. If t is not a $\perp$-tuple, then we add the tuple t with local condition $\lambda^{\mathcal{T}}(t)$ to $R^{\mathcal{T}}$, where $R^{\mathcal{T}}$ is the corresponding table from the c-multitable. The local condition $\lambda^{\mathcal{T}}(t)$ is defined as

$$\lambda^{\mathcal{T}}(t) = \begin{cases} \mathit{true} & n_j = 1 \\ (x_j = i) & 1 \leq i < n_j \\ (x_j \neq 1 \wedge \ldots \wedge x_j \neq n_j - 1) & 1 < i = n_j. \end{cases}$$

   Here x_j is a new variable for the component C_j not occurring in $\mathcal{W}$, which encodes to which row of component C_j a tuple belongs to.

Correctness. Let $\mathbf{A}$ be the g-tabset over relational schema $(R_1[U_1], \ldots, R_k[U_k])$ encoded by $\mathcal{W}$. We denote our translation by f. We next prove that the translation is correct, that is,

$$\mathit{rep}\,(\mathcal{W}) = \mathit{rep}\,(f(\mathcal{W}))$$

Let $\mathcal{T} = f(\mathcal{W}) = (R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$ be the resulting x-multitable with global condition $\phi^{\mathcal{T}}$ and a mapping $\lambda^{\mathcal{T}}$ from the tuples to their local conditions. We show that

$$\mathcal{A} \in \mathit{rep}\,(\mathcal{W}) \Leftrightarrow \mathcal{A} \in \mathit{rep}(\mathcal{T})$$

The case when $\mathcal{W}$ has empty components follows immediately from the definition of gWSDs and unsatisfiable x-tables. We consider next the case of non-empty components.

Let T be a set of tuple identifiers for a table R. We denote by $\theta^{\mathcal{A}}$ the partial injective function from tuple identifiers to tuples in $\mathcal{A}$. For each g-multitable $A \in \mathbf{A}$ and tuple identifier $d \in T$, if defined, $\theta^A(d)$ is a tuple of the schema of R, possibly with variables.

Let $x_1, \ldots, x_m$ be the variables for components $C_1, \ldots, C_m$, respectively, that were created by the translation f.

1) Let $\mathcal{A} \in rep(\mathcal{W})$. Then there exists a valuation ν consistent with ϕ and tuples $w_{i_j} \in C_j$, such that $\mathcal{A} = \nu(\text{inline}^{-1}(w_i))$, where $w_i = w_{i_1} \times \ldots \times w_{i_m}$.

Let ν' be the valuation ν extended with the mappings $x_j \mapsto i_j$. Let $\mathcal{A}' = \nu'(\mathcal{T})$.

Let d be a tuple identifier for a relation schema $R[U]$ and let $R^{\mathcal{T}}$ be the image of R under f. If $t = \theta^{\text{inline}^{-1}(w_i)}(d)$ is defined, then $\nu(t) \in R^{\mathcal{A}}$. But then $f(t)$ is defined and let the tuple t' with local condition $\lambda^{\mathcal{T}}(t')$ be the image of t under f. From the construction of ν' we have $\nu'(t') = \nu(t)$ and $\nu'(\lambda^{\mathcal{T}}(t')) = true$. But then $\nu'(t') \in R^{\nu'(\mathcal{T})}$ and hence $\nu(t) \in R^{\nu'(\mathcal{T})}$. If for a tuple $t' \in R^{\mathcal{T}}$ we have $\nu'(t') \in R^{\nu'(\mathcal{T})}$, then for the prototype t of t' we have $\nu(t) = \nu'(t')$ and $\nu(t) \in R^{\mathcal{A}}$ and hence $\nu'(t') \in R^{\mathcal{A}}$. It follows $\mathcal{A} = \mathcal{A}'$.

2) Let now $\mathcal{A} \in rep(\mathcal{T})$. Then there exists a valuation ν consistent with $\phi^{\mathcal{T}}$ such that $\mathcal{A} = \nu(\mathcal{T})$ and if t is a tuple in a table $R^{\mathcal{T}}$ with local condition $\lambda^{\mathcal{T}}(t)$

$$\nu(t) \in R^{\nu(\mathcal{T})} \text{ iff } \nu(\lambda^{\mathcal{T}}(t)) = true$$

For $1 \leq j \leq m$ if variable x_j exists, let $i_j := \nu(x_j)$. Let w_{i_j} be the tuple $w_{i_j} \in C_j$ if $1 \leq i_j \leq n_j$, otherwise let $w_{i_j} = w_{n_j}$. If x_j does not exist, then by our construction component C_j contained a single tuple and let $w_{i_j} := w_1$ be that tuple. Let $w = w_{i_1} \times \ldots \times w_{i_m}$. Let $\mathcal{A}' = \nu(w)$. For a tuple $t \in R^{\mathcal{T}}$ we have $\nu(t) \in R^{\nu(\mathcal{T})}$ iff $\nu(\lambda^{\mathcal{T}}(t)) = true$. If $\nu(t) \in R^{\nu(\mathcal{T})}$, let t' be the prototype of t in w. By construction we have $\nu(t) = \nu(t')$ and $\nu(t) \in R^{\mathcal{A}'}$. By analogy, if for a tuple identifier d the image $t = \theta^{\text{inline}^{-1}(w)}(d)$ is defined, then $\nu(t) \in R^{\mathcal{A}'}$. But then for the image $t' = f(t)$ we have $\nu(\lambda^{\mathcal{T}}(t')) = true$ and $\nu(t') \in R^{\nu(\mathcal{T})}$. Hence $\mathcal{A} = \mathcal{A}'$. □

Example 4. Consider the 1-gWSD $(\{C_1\}, \phi)$ given in Figure 5(a). The first tuple of C_1 encodes a g-table R with a single tuple (with identifier d_1), and the second tuple of C_1 encodes two v-tuples with identifiers d_1 and d_2. The encoding of C_1 as an x-table T with global condition $\phi^{\mathcal{T}}$ is given in Figure 5(b). □

C_1	$R.d_1.A$	$R.d_1.B$	$R.d_2.A$	$R.d_2.B$
	x	y	$\bot$	$\bot$
	1	z	z	3

$\phi = (x \neq 1) \wedge (x \neq y) \wedge (z \neq 2)$

(a) 1-gWSD

T	A	B	$cond$
			$\phi^{\mathcal{T}} = (x \neq 1) \wedge (x \neq y) \wedge (z \neq 2)$
	x	y	$(x_1 = 1)$
	1	z	$(x_1 \neq 1)$
	z	3	$(x_1 \neq 1)$

(b) x-table equivalent to the 1-gWSD (a)

Fig. 5. Translating gWSDs into x-multitables.

For the other, somewhat more involved direction,

Lemma 2. *Any c-multitable can be represented by an equivalent gWSD.*

Proof. For simplicity we show the construction for c-tables but it can be straightforwardly generalized for c-multitables. Let $\mathcal{T} = (T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$ be a c-table with global condition $\phi^{\mathcal{T}}$ and a mapping $\lambda^{\mathcal{T}}$ from the tuples to their local conditions. Without loss of generality we assume that all variables in the c-table are different.

$T^{\mathcal{T}}$	A_1	$\dots$	A_k	*cond*
				$\phi^{\mathcal{T}}$
d_1	$x_{1,1}$	$\dots$	$x_{1,k}$	$\lambda_1^{\mathcal{T}}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
d_n	$x_{n,1}$	$\dots$	$x_{n,k}$	$\lambda_n^{\mathcal{T}}$

Let $\mathbf{X}_T$ and $\mathbf{D}_T$ be the set of all variables and the set of all constants appearing in the c-table, respectively.

Construction. We define a corresponding 1-gWSD $(\{C\}, \phi')$ with gTST C of schema

$$C(d_1.A_1, \dots, d_1.A_k, \dots, d_n.A_1, \dots, d_n.A_k)$$

as follows. We consider comparisons of the form $\tau = \tau'$ and $\tau \neq \tau'$ where $\tau, \tau' \in \mathbf{X}_T \cup \mathbf{D}_T$ are variables or constants from the c-table. We compute a set of consistent $\Theta = \bigwedge\{\tau\, \theta_{\tau,\tau'}\, \tau' \mid \tau, \tau' \in \mathbf{X}_T \cup \mathbf{D}_T\}$ where $\theta_{\tau,\tau'} \in \{=, \neq\}$ for all τ, τ' and $\Theta \vDash \phi^{\mathcal{T}}$. Note that the equalities in Θ define an equivalence relation on $\mathbf{X}_T \cup \mathbf{D}_T$. In particular, we take into account that $c = c'$ is consistent iff c and c' are the same constant. We denote by $[x_{i,j}]_=$ the equivalence class of a variable $x_{i,j}$ with respect to the equalities given by Θ and by $h([x_{i,j}]_=)$ the representative element of that equivalent class (e.g. the first element with respect to any fixed order of the elements in the class).

We have tuple $\langle s_{1,1}, \dots, s_{1,k}, \dots, s_{n,1}, \dots, s_{n,k}\rangle \in C$ if and only if for some consistent Θ

$$s_{i,j} = \begin{cases} \bot & \dots\ \Theta \nvDash \lambda_i^{\mathcal{T}} \\ c & \dots\ c \in \mathbf{D}_T, \Theta \vDash \lambda_i^{\mathcal{T}}, \Theta \vDash (x_{i,j} = c) \\ h([x_{i,j}]_=) & \dots\ \Theta \vDash \lambda_i^{\mathcal{T}}, \forall c \in \mathbf{D}_T\ \Theta \vDash (x_{i,j} \neq c) \end{cases}$$

To construct the global condition for the gWSD, we create a conjunction over the negated equalities appearing in the Θ-s from the previous step.

Let $\mathcal{T} = (T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$ be a c-table with global condition $\phi^{\mathcal{T}}$ and a mapping $\lambda^{\mathcal{T}}$ from the tuples to their local conditions and let the 1-gWSD $(\{C\}, \phi')$ be the result of translating the given c-table.

Correctness. We next show that:

$$\mathcal{A} \in rep(T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}}) \Leftrightarrow \mathcal{A} \in rep(\{C\}, \phi')$$

(1) Given any world $\mathcal{A} \in rep(T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$. Then there exist an injective function f mapping each tuple of $\mathcal{A}$ to a distinct tuple of the c-table, and a consistent Θ such that

(a) $\Theta \models \phi^{\mathcal{T}}$,
(b) for all tuples t_i of the c-table not in the image of f, $\Theta \not\models \lambda_i^{\mathcal{T}}$,
(c) for all tuples $t_i = f(t')$ of the c-table for some tuple t' of $\mathcal{A}$, $\Theta \models \lambda_i^{\mathcal{T}}$ and $x_{i,j} = s_j$, and
(d) for all $\tau, \tau' \in (\mathbf{X}_T \cup \mathbf{D}_T)$, either $\tau = \tau'$ or $\tau \neq \tau'$.

But then, it is immediate that by our construction there is a tuple $w = \langle s_{1,1}, \ldots, s_{1,k}, \ldots, s_{n,1}, \ldots, s_{n,k} \rangle \in C$ such that $s_{i,j} = c_{i,j}$ iff there is a t' in $\mathcal{A}$ with $f(t') = t_i$ and $s_j = c_{i,j}$, and $s_{i,\cdot} = \bot$ iff there is no t' in $\mathcal{A}$ with $f(t') = t'_i$. Thus $\mathcal{A} \in rep(w)$ and therefore $\mathcal{A} \in rep(\{C\}, \phi')$.

(2) Given any world $\mathcal{A} \in rep(\{C\}, \phi')$. Then there is a tuple

$$w = \langle s_{1,1}, \ldots, s_{1,k}, \ldots, s_{n,1}, \ldots, s_{n,k} \rangle \in C$$

that represents $\mathcal{A}$ with an associated Θ.

Let Θ' be the consistent conjunction that extends Θ by the equalities $x_{i,j} = a_{i,j}$ given by $\mathcal{A}$ and defines an equivalence relation on $\mathbf{X}_T \cup \mathbf{D}_T \cup |\mathcal{A}|$. Since tuple w of C represents $\mathcal{A}$, for all $s_{i,j} = x_{i,j}$, $(\Theta \wedge \phi^{\mathcal{T}}) \not\models x_{i,j} \neq a_{i,j}$. Thus such a consistent Θ' exists.

It follows that $\Theta' \models \phi^{\mathcal{T}}$ and that $\Theta' \models \lambda_i^{\mathcal{T}}$ if and only if $\Theta \models \lambda_i^{\mathcal{T}}$. But then $\mathcal{A} \in rep(T, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$. □

Example 5. Consider the c-table T with global condition ϕ of Figure 6(a). The equivalent gWSD $(\{C\}, \phi')$ is given in Figure 6(b). □

T	A	B	*cond*
			$\phi = (x \neq 1) \wedge (x = z)$
d_1	x	1	$(x \neq 2)$
d_2	z	y	$(y \neq 2)$

(a) c-table T with global condition ϕ

C	$R.d_1.A$	$R.d_1.B$	$R.d_2.A$	$R.d_2.B$
	$\bot$	$\bot$	$\bot$	$\bot$
	$\bot$	$\bot$	2	y
	x	1	$\bot$	$\bot$
	x	1	x	y

$\phi' = (x \neq 1) \wedge (x \neq 2) \wedge (y \neq 2)$

(b) Equivalent 1-gWSD $(\{C\}, \phi')$ for the c-table of Figure 6(a).

Fig. 6. Translating c-tables into gWSDs.

As a corollary of Lemma 1, it follows that x-multitables, a syntactically restricted form of c-multitables, are at least as expressive as gWSDs. However, by Lemma 2, gWSDs are at least as expressive as c-multitables. This implies that

Corollary 5. *The x-multitables capture gWSDs and thus c-multitables.*

In the next section we will use the linear-time translation of gWSDs into x-multitables to characterize the data complexity of the tuple q-possibility problem for gWSDs and positive relational algebra.

C_1	$R.d_1.A$	$R.d_2.A$	$S.d_1.B$
	2	y	z
	$\perp$	2	2

C_2	$R.d_3.A$
	1

C_3	$S.d_2.B$
	1
	2

(a) 3-gWSD $\mathcal{W} = (\{C_1, C_2, C_3\}, \mathit{true})$

R	A	cond
		true
	2	$x_1 = 1$
	y	$x_1 = 1$
	2	$x_1 \neq 1$
	1	true

S	B	cond
		true
	z	$x_1 = 1$
	2	$x_1 \neq 1$
	1	$x_3 = 1$
	2	$x_3 \neq 1$

(b) x-multitable $T = (R, S)$

Fig. 7. Example of a 3-gWSD and an equivalent x-multitable.

5 Complexity of Managing gWSDs

We consider the data complexity of the decision problems defined in Section 1. Note that in the literature the tuple (q-)possibility and (q-)certainty problems are sometimes called bounded or restricted (q-)possibility, and (q-)certainty respectively, and the instance (q-)possibility and (q-)certainty are sometimes called (q-)membership and (q-)uniqueness [3]. A comparison of the complexity results for these decision problems in the context of gWSDs to those of c-tables [3] and Trio [8] is given in Table 2.

We first prove complexity results for tuple q-possibility in the context of x-tables. This is particularly relevant as gWSDs can be translated in linear time into x-tables, as done in the proof of Lemma 1.

Lemma 3. *Tuple q-possibility is in PTIME for x-tables and positive relational algebra.*

Proof. Recall from Definition 2 and Proposition 3 that x-tables are closed under positive relational algebra and the evaluation of positive relational algebra queries on x-tables is in PTIME.

Consider a constant tuple t and a fixed positive relational query Q, both over schema U, and two x-multitables $\mathcal{T}$ and $\mathcal{T}'$ such that $\mathcal{T}' = Q(\mathcal{T})$.

In case the global condition of $\mathcal{T}'$ is unsatisfiable, then $\mathcal{T}'$ represents the empty world-set and t is not possible. The global condition is a conjunction of negated equalities and we can check its unsatisfiability in PTIME. We consider next the case of satisfiable global conditions. Following the semantics of x-tables, the tuple t is possible in $\mathcal{T}'$ iff there is a tuple t' in $\mathcal{T}'$ and a valuation ν consistent with the global and local conditions such that t' equals t under ν. This can be checked for each $\mathcal{T}'$-tuple individually and in PTIME. □

Theorem 3. *Tuple q-possibility is in PTIME for gWSDs and positive relational algebra.*

Proof. This follows from the linear time translation of gWSDs into x-multitables ensured by Lemma 1 and the PTIME result for x-multitables given in Lemma 3.□

For full relational algebra, tuple q-possibility becomes NP-hard even for v-tables where each variable occurs at most once (also called Codd tables) [3].

Theorem 4. *Tuple q-possibility is in NP for gWSDs and relational algebra and NP-hard for WSDs and relational algebra.*

Proof. Tuple q-possibility is in NP for gWSDs and relational algebra because gWSDs can be translated linearly into c-tables (see Lemma 1) and tuple q-possibility is in NP for c-tables and full relational algebra [3].

We show NP-hardness for WSDs and relational algebra by a reduction from 3CNF-satisfiability [15]. Given a set $\mathbf{Y}$ of propositional variables and a set of clauses $c_i = c_{i,1} \vee c_{i,2} \vee c_{i,3}$ such that for each i, k, $c_{i,k}$ is x or $\neg x$ for some $x \in \mathbf{Y}$, the 3CNF-satisfiability problem is to decide whether there is a satisfying truth assignment for $\bigwedge_i c_i$.

Construction. We create a WSD $\mathcal{W} = (C_1, \ldots, C_{|\mathbf{Y}|}, C_S)$ representing worlds of two relations R and S over schemas $R(C)$ and $S(C)$, respectively, as follows[6]. For each variable x_i in $\mathbf{Y}$ we create a component C_i with two local worlds, one for x_i and the other for $\neg x_i$. For each literal $c_{i,k}$ we create an R-tuple $\langle i \rangle$ with id $d_{i,k}$. In case $c_{i,k} = x_j$ or $c_{i,k} = \neg x_j$, then the schema of C_j contains the attribute $R.d_{i,k}.C$ and the local world for x_j or $\neg x_j$, respectively, contains the values $\langle i \rangle$ for these attributes. All component fields that remained unfilled are finally filled in with $\bot$-values. The additional component S_C has n attributes $S.d_1.C, \ldots, S.d_n.C$ and one local world $(1, \ldots, n)$. Thus, by construction, $S = \{\langle 1 \rangle, \ldots, \langle n \rangle\}$ and $R \subseteq S$ in all worlds defined by $\mathcal{W}$.

The problem of deciding whether $\bigwedge_i c_i$ has a satisfying truth assignment is equivalent to deciding whether there is a world $\mathcal{A} \in \mathit{rep}(\mathcal{W})$ such that the answer to the query $S - R$ is empty.

Correctness. We prove the correctness of the reduction by showing that $\bigwedge_i c_i$ has a satisfying truth assignment exactly when $\exists \mathcal{A} \in \mathit{rep}(\mathcal{W}) : S^{\mathcal{A}} - R^{\mathcal{A}} = \emptyset$. Because $\forall \mathcal{A} \in \mathit{rep}(\mathcal{W}) : R^{\mathcal{A}} \subseteq S^{\mathcal{A}}$, the condition is equivalent to $\exists \mathcal{A} \in \mathit{rep}(\mathcal{W}) : S^{\mathcal{A}} = R^{\mathcal{A}}$.

First, assume there is a truth assignment ν of $\mathbf{Y}$ that proves $\bigwedge_i c_i$ is satisfiable. Then, $\nu(c_i)$ is *true* for each clause c_i. Because each clause c_i is a disjunction, this means there is at least one $c_{i,k}$ for each c_i such that $\nu(c_{i,k})$ is *true*.

Turning to $\mathcal{W}$, ν represents a choice of local worlds of $\mathcal{W}$ such that for each variable $x_j \in \mathbf{Y}$ if $\nu(x_j) = \mathit{true}$ then we choose the first local world of C_j and if $\nu(x_j) = \mathit{false}$ then we choose the second local world of C_j. Let w_j be the choice for C_j and let w_{C_S} be the only choice for C_S. Then, $\mathcal{W}$ defines a world $\mathcal{A} = \mathrm{inline}^{-1}(w_1 \times \ldots \times w_{|\mathbf{Y}|} \times w_{C_S})$ and $R^{\mathcal{A}}$ contains those tuples defined in the chosen local worlds. Because there is at least one $c_{i,k}$ per clause c_i such that $\nu(c_{i,k})$ is *true*, there is also a local world w_j that defines R-tuple $\langle i \rangle$ for each c_i. Thus $R^{\mathcal{A}} = S^{\mathcal{A}}$.

Now, assume there exists a world $\mathcal{A} \in \mathit{rep}(\mathcal{W})$ such that $S^{\mathcal{A}} = R^{\mathcal{A}}$. Thus $R^{\mathcal{A}} = \{\langle 1 \rangle, \ldots, \langle n \rangle\}$ and there is a choice of local worlds of the components in $\mathcal{W}$ that define all R-tuples $\langle 1 \rangle$ through $\langle n \rangle$. By construction, this choice corresponds to a truth assignment ν that maps at least one literal $c_{i,k}$ of each c_i to *true*. Thus ν is a satisfying truth assignment of $\bigwedge_i c_i$. □

Example 6. Figure 8 gives a 3CNF clause set and its WSD encoding. Checking the satisfiability of $c_1 \wedge c_2 \wedge c_3$ is equivalent to checking whether there is a choice of

[6] For clarity reasons, we use two relations; they can be represented as one relation with an additional attribute stating the relation name.

3CNF clause set: $\{c_1 = x_1 \vee x_2 \vee x_3, \quad c_2 = x_1 \vee \neg x_2 \vee x_4, \quad c_3 = \neg x_1 \vee x_2 \vee \neg x_4\}$

C_1	$R.d_{1,1}$.C	$R.d_{2,1}$.C	$R.d_{3,2}$.C
(x_1)	1	2	$\bot$
$(\neg x_1)$	$\bot$	$\bot$	3

C_2	$R.d_{1,2}$.C	$R.d_{2,2}$.C	$R.d_{3,2}$.C
(x_2)	1	$\bot$	3
$(\neg x_2)$	$\bot$	2	$\bot$

C_3	$R.d_{1,3}$.C
(x_3)	1
$(\neg x_3)$	$\bot$

C_4	$R.d_{2,3}$.C	$R.d_{3,3}$.C
(x_4)	2	$\bot$
$(\neg x_4)$	$\bot$	3

C_S	$S.d_1$.C	$S.d_2$.C	$S.d_3$.C
w_{C_S}	1	2	3

Fig. 8. 3CNF clause set encoded as tuple q-possibility problem for WSDs.

local worlds in the WSD such that $S - R$ is empty. This also means that R should contain the tuples $\langle 1 \rangle$ through $\langle 3 \rangle$. For example, $x_1 \mapsto \mathit{true}, x_2 \mapsto \mathit{true}, x_3 \mapsto \mathit{true}, x_4 \mapsto \mathit{true}$ is a satisfying truth assignment. Indeed, the corresponding choice of local worlds $(C_1 : x_1, C_2 : x_2, C_3 : x_3, C_4 : x_4, C_S : w_{C_S})$ defines a world $\mathcal{A}$ in which $R^{\mathcal{A}}$ equals $S^{\mathcal{A}}$. □

Theorem 5. *Tuple possibility and certainty are in PTIME for gWSDs.*

Proof. The result for tuple possibility follows directly from Theorem 3, where the positive relational query is the identity.

For tuple certainty, consider a tuple-level gWSD $W = (\{C_1, \ldots, C_m\}, \phi)$ and a tuple t. Tuple t is certain exactly if ϕ is unsatisfiable or there is a component C_i such that each tuple of C_i contains t (without variables). Suppose ϕ is satisfiable and for each component C_i there is at least one tuple $w_i \in C_i$ that does not contain t. Then there is a world-tuple $w \in C_1 \times \cdots \times C_m$ such that tuple t does not occur in w. If there is a mapping θ that maps some tuple in w to t and for which $\theta(\phi)$ is true, then there is also a mapping θ' such that $\theta'(w)$ does not contain t but $\theta'(\phi)$ is true. Thus t is not certain. □

Theorem 6. *Instance possibility is in NP for gWSDs and NP-hard for WSDs.*

Proof. Let $\mathcal{W} = (\{C_1, \ldots, C_n\}, \phi)$ be a gWSD. The problem is in NP since we can nondeterministically check whether there is a choice of tuples $t_1 \in C_1, \ldots, t_n \in C_n$ such that $t_1 \circ \cdots \circ t_n$ represents the input instance.

We show NP-hardness for WSDs with a reduction from Exact Cover by 3-Sets [15].

Given a set X with $|X| = 3q$ and a collection C of 3-element subsets of X, the exact cover by 3-sets problem is to decide whether there exists a subset $C' \subseteq C$, such that every element of X occurs in exactly one member of C'.

Construction. The set X is encoded as an instance consisting of a unary relation I_X over schema $I_X[A]$ with $3q$ tuples. The collection C is represented as a WSD $\mathcal{W} = \{C_1, \ldots, C_q\}$ encoding a relation R over schema $R[A]$, where $C_1, \ldots, C_q$ are component relations. The schema of a component C_i is $C_i[R.d_{j+1}.A, R.d_{j+2}.A, R.d_{j+3}.A]$, where $j = \lfloor \frac{i}{3} \rfloor$. Each 3-element set $c = \{x, y, z\} \in C$ is encoded as a tuple (x, y, z) in each of the components C_i.

I_X	A
	1
	2
	3
	4
	5
	6
	7
	8
	9

C_1	t_1.A	t_2.A	t_3.A
w_1	1	5	9
w_2	2	5	8
w_3	3	4	6
w_4	2	7	8
w_5	1	6	9

C_2	t_4.A	t_5.A	t_6.A
w_1	1	5	9
w_2	2	5	8
w_3	3	4	6
w_4	2	7	8
w_5	1	6	9

C_3	t_7.A	t_8.A	t_9.A
w_1	1	5	9
w_2	2	5	8
w_3	3	4	6
w_4	2	7	8
w_5	1	6	9

Fig. 9. Exact cover by 3-sets encoded as instance possibility problem for WSDs.

The problem of deciding whether there is an exact cover by 3-sets of X is equivalent to deciding whether $I_X \in \mathit{rep}(\mathcal{W})$.

Correctness. We prove the correctness of the reduction, that is, we show that X has an exact cover by 3-sets exactly when $I_X \in \mathit{rep}(\mathcal{W})$.

First, assume there is a world $\mathcal{A} \in \mathit{rep}(\mathcal{W})$ with $R^{\mathcal{A}} = I_X$. Then there exist tuples $w_i \in C_i, 1 \leq i \leq q$, such that $\mathcal{A} = \mathit{rep}(\{w_1\} \times \ldots \times \{w_q\})$. As I_X and $R^{\mathcal{A}}$ have the same number of tuples and all elements of I_X are different, it follows that the values in $w_1, \ldots, w_q$ are disjoint. But then this means that the elements in $w_1, \ldots, w_q$ are an exact cover of X.

Now, assume there exists an exact cover $C' = \{c_1, \ldots, c_q\}$ of X. Let $w_i \in C_i$ such that $w_i = c_i, 1 \leq i \leq q$. As the elements c_i are disjoint, the world $\mathcal{A} = \mathit{rep}(\{w_1\} \times \ldots \times \{w_q\})$ contains exactly $3q$ tuples. Since C' is an exact cover of X and each element of X (and therefore of I_X) appears in exactly one local world w_i, it follows that $I_X = R^{\mathcal{A}}$. □

Example 7. Consider the set X and the collection of 3-element sets C defined as

$$X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$C = \{\{1, 5, 9\}, \{2, 5, 8\}, \{3, 4, 6\}, \{2, 7, 8\}, \{1, 6, 9\}\}$$

The encoding of X and C is given in Figure 9 as WSD $\mathcal{W}$ and instance I_X. A possible cover of X, or equivalently, a world of $\mathit{rep}(\mathcal{W})$ equivalent to I_X, is the world $\text{inline}^{-1}(w_1 \circ w_3 \circ w_4)$ or, by resolving the record composition,

$$\text{inline}^{-1}(t_1.A : 1, t_2.A : 5, t_3.A : 9, t_4.A : 3, t_5.A : 4, t_6 : A : 6, t_7 : 2, t_8 : 7, t_9.A : 8).$$

□

Theorem 7. *Instance certainty is in PTIME for gWSDs.*

Proof. Given an instance I and a gWSD W representing a relation R, the problem is equivalent to checking for each world $\mathcal{A} \in \mathit{rep}(W)$ whether (1) $I \subseteq R^{\mathcal{A}}$ and (2) $R^{\mathcal{A}} \subseteq I$. Test (1) is reducible to checking whether each tuple from I is certain in R, and is thus in PTIME (cf. Theorem 5). For (2), we check in PTIME whether there is a non $\perp$-tuple in some world of $\mathit{rep}(W)$ that is not in the instance I. If W has variables then it cannot represent certain instances. □

Theorem 8. *Instance q-possibility is NP-complete for gWSDs and relational algebra.*

Proof. For the identity query, the problem becomes instance possibility, which is NP-complete (see Theorem 6). To show it is in NP, we use the PTIME reduction from gWSDs to c-tables given in Lemma 1 and the NP-completeness result for instance q-possibility and c-tables [3]. □

Theorem 9. *Tuple and instance q-certainty are in coNP for gWSDs and relational algebra and coNP-hard for WSDs and positive relational algebra.*

Proof. Tuple and instance q-certainty are in coNP for gWSDs and positive relational algebra because gWSDs can be translated linearly into c-tables (see Lemma 1) and tuple and instance q-certainty are in coNP for c-tables and full relational algebra [3].

We show coNP-hardness for WSDs and positive relational algebra by a reduction from 3DNF-tautology [15]. Given a set $\mathbf{Y}$ of propositional variables and a set of clauses $c_i = c_{i,1} \wedge c_{i,2} \wedge c_{i,3}$ such that for each i, k, $c_{i,k}$ is x or $\neg x$ for some $x \in \mathbf{Y}$, the 3DNF-tautology problem is to decide whether $\bigvee_i c_i$ is true for each truth assignment of $\mathbf{Y}$.

Construction. We create a WSD $\mathcal{W} = (C_1, \ldots, C_{|\mathbf{Y}|})$ representing worlds of a relation R over schema $R(C, P)$ as follows. For each variable x_i in $\mathbf{Y}$ we create a component C_i with two local worlds, one for x_i and the other for $\neg x_i$. For each literal $c_{i,k}$ we create an R-tuple (i, k) with id $d_{i,k}$. In case $c_{i,k} = x_j$ or $c_{i,k} = \neg x_j$, then the schema of C_j contains the attributes $R.d_{i,k}.C$ and $R.d_{i,k}.P$, and the local world for x_j or $\neg x_j$, respectively, contains the values (i, k) for these attributes. All component fields that remained unfilled are finally filled in with $\perp$-values.

The problem of deciding whether $\bigvee_i c_i$ is a tautology is equivalent to deciding whether the empty tuple $\langle\rangle$ is certain in the answer to the fixed positive relational algebra query $Q := \pi_\emptyset(\sigma_\phi(R\ r_1 \times R\ r_2 \times R\ r_3))$, where

$$\phi := (r_1.C = r_2.C \text{ and } r_1.C = r_3.C \text{ and } r_1.P = 1 \text{ and } r_2.P = 2 \text{ and } r_3.P = 3).$$

Correctness. We prove the correctness of the reduction, that is, we show that $\bigvee_i c_i$ is a tautology exactly when $\forall \mathcal{A} \in \mathit{rep}(\mathcal{W}) : \langle\rangle \in R^{\mathcal{A}}$.

First, assume there is a truth assignment ν of $\mathbf{Y}$ that proves $\bigvee_i c_i$ is not a tautology. Then, there exists a choice of local worlds of $\mathcal{W}$ such that for each variable $x_i \in \mathbf{Y}$ if $\nu(x_i) = \mathit{true}$ then we choose the first local world of C_i and if $\nu(x_i) = \mathit{false}$ then we choose the second local world of C_i. Let w_i be the choice for C_i. Then, $\mathcal{W}$ defines a world $\mathcal{A} = \text{inline}^{-1}(w_1 \times \ldots \times w_{|\mathbf{Y}|})$ and $R^{\mathcal{A}}$ contains those tuples defined in the chosen local worlds. If ν proves $\bigvee_i c_i$ is not a tautology, then $\nu(\bigvee_i c_i)$ is *false* and, because $\bigvee_i c_i$ is a disjunction, no clause c_i is *true*. Thus $R^{\mathcal{A}}$ does not contain tuples $(i, 1)$, $(i, 2)$, and $(i, 3)$ for each clause c_i. This means that the condition of Q cannot be satisfied and thus the answer of Q is empty. Thus the tuple $\langle\rangle$ is not certain in the answer to Q.

Now, assume there exists a world $\mathcal{A} \in \mathit{rep}(\mathcal{W})$ such that $\langle\rangle \notin R^{\mathcal{A}}$. Then, $R^{\mathcal{A}}$ contains no triple $(i, 1)$, $(i, 2)$, and $(i, 3)$ for any clause c_i, because such a triple

3DNF clause set: $\{c_1 = x_1 \wedge x_2 \wedge x_3, \quad c_2 = x_1 \wedge \neg x_2 \wedge x_4, \quad c_3 = \neg x_1 \wedge x_2 \wedge \neg x_4\}$

C_1	$R.d_{1,1}$.(C,P)	$R.d_{2,1}$.(C,P)	$R.d_{3,2}$.(C,P)
(x_1)	(1,1)	(2,1)	$\bot$
$(\neg x_1)$	$\bot$	$\bot$	(3,2)

C_3	$R.d_{1,3}$.(C,P)
(x_3)	(1,3)
$(\neg x_3)$	$\bot$

C_2	$R.d_{1,2}$.(C,P)	$R.d_{2,2}$.(C,P)	$R.d_{3,1}$.(C,P)
(x_2)	(1,2)	$\bot$	(3,1)
$(\neg x_2)$	$\bot$	(2,2)	$\bot$

C_4	$R.d_{2,3}$.(C,P)	$R.d_{3,3}$.(C,P)
(x_4)	(2,3)	$\bot$
$(\neg x_4)$	$\bot$	(3,3)

Fig. 10. 3DNF clause set encoded as tuple and instance q-certainty problem for WSDs.

satisfies the selection condition. This means that the choice of local worlds of the components in $\mathcal{W}$ correspond to a valuation ν that does not map all $c_{i,1}$, $c_{i,2}$, and $c_{i,3}$ to true, for any clause c_i. Thus $\bigvee_i c_i$ is not a tautology.

Because by construction $R^{\mathcal{A}}$ is either $\{\}$ or $\{\langle\rangle\}$ for any world $\mathcal{A} \in \mathit{rep}(\mathcal{W})$, the same proof works also for instance q-certainty with instance $\{\langle\rangle\}$. □

Example 8. Figure 10 gives a 3DNF clause set and its WSD encoding. Checking tautology of $H := c_1 \vee c_2 \vee c_3$ is equivalent to checking whether the empty tuple is certain in the answer to the query from the proof of Theorem 9. Formula H is not a tautology because it becomes *false* under the truth assignment $\{x_1 \mapsto \mathit{true}, x_2 \mapsto \mathit{true}, x_3 \mapsto \mathit{false}, x_4 \mapsto \mathit{true}\}$. This is equivalent to checking whether the empty tuple is in the answer to our query in the world $\mathcal{A}$ defined by the first local world of C_1 (encoding $x_1 \mapsto \mathit{true}$), the first local world of C_2 (encoding $x_2 \mapsto \mathit{true}$), the second local world of C_3 (encoding $x_3 \mapsto \mathit{false}$), and the first local world of C_4 (encoding $x_4 \mapsto \mathit{true}$). The relation $R^{\mathcal{A}}$ contains the tuples $\{(1,1),(2,1),(1,2),(3,1),(2,3)\}$ and the query answer is empty. □

6 Optimizing gWSDs

In this section we study the problem of optimizing a given gWSD by further decomposing its components using the product operation. We note that product decomposition corresponds to the notion of *relational factorization*. We then define this new notion and study some of its properties, like uniqueness and primality or minimality. It turns out that any relation admits a unique minimal factorization, and there is an algorithm that can compute it efficiently. Because gWSD components are special relations with variables and the $\bot$-symbol, they can admit several minimal factorizations and our efficient algorithm can not always find one of them (but it can still find good non-optimal factorizations by treating variables as constants). However, the (tuple-level) WSDs admit prime factorizations that are unique modulo the $\bot$-symbol[7] and can be efficiently computed by a trivial extension of our algorithm with the tuple-level constraint.

[7] Two tuples $(A_1 : \bot, \ldots, A_n : \bot)$ and $(A_1 : a_1, \ldots, A_n : a_n)$ of a relation defined by a (g)WSD, where at least one a_i is $\bot$, are equivalent modulo the $\bot$-symbol.

6.1 Prime Factorizations of Relations

Definition 6. Let there be schemata $R[U]$ and $Q[U']$ such that $\emptyset \subset U' \subseteq U$. A *factor* of a relation R over schema $R[U]$ is a relation Q over schema $Q[U']$ such that there exists a relation R' with $R = Q \times R'$.

A factor Q of R is called *proper*, if $Q \neq R$. A factor Q is *prime*, if it has no proper factors. Two relations over the same schema are *coprime*, if they have no common factors.

Definition 7. Let R be a relation. A *factorization* of R is a set $\{C_1, \ldots, C_n\}$ of factors of R such that $R = C_1 \times \ldots \times C_n$.

In case the factors $C_1, \ldots, C_n$ are prime, the factorization is said to be *prime*. From the definition of relational product and factorization, it follows that the schemata of the factors $C_1, \ldots, C_n$ are a disjoint partition of the schema of R.

Proposition 9. *For each relation a prime factorization exists and is unique.*

Proof. Consider any relation R. Existence is clear because R admits the factorization $\{R\}$, which is prime in case R is prime.

Uniqueness is next shown by contradiction. Assume R admits two different prime factorizations $\{\pi_{U_1}(R), \ldots, \pi_{U_m}(R)\}$ and $\{\pi_{V_1}(R), \ldots, \pi_{V_m}(R)\}$. Since the two factorizations are different, there are two sets U_i, V_j such that $U_i \neq V_j$ and $U_i \cap V_j \neq \emptyset$. But then, as of course $R = \pi_{U-V_j}(R) \times \pi_{V_j}(R)$, we have $\pi_{U_i}(R) = \pi_{U_i}\big(\pi_{U-V_j}(R) \times \pi_{V_j}(R)\big) = \pi_{U_i-V_j}(R) \times \pi_{U_i \cap V_j}(R)$. It follows that $\{\pi_{U_1}(R), \ldots, \pi_{U_{i-1}}(R), \pi_{U_i-V_j}(R), \pi_{U_i \cap V_j}(R), \pi_{U_{i+1}}(R), \ldots, \pi_{U_m}(R)\}$ is a factorization of R, and the initial factorizations cannot be prime. Contradiction. □

6.2 Computing Prime Factorizations

This section first gives two important properties of relational factors and factorizations. Based on them, it further devises an efficient yet simple algorithm for computing prime factorizations.

Proposition 10. *Let there be two relations S and F, an attribute A of S and not of F, and a value $v \in \pi_A(S)$. Then, for some relations R, E, and I holds*

$$S = F \times R \Leftrightarrow \sigma_{A=v}(S) = F \times E \text{ and } \sigma_{A \neq v}(S) = F \times I.$$

Proof. Note that the schemata of F and R represent a disjoint partition of the schema of S and thus A is an attribute of R.

$\Rightarrow$. Relation F is a factor of $\sigma_{A=v}(S)$ because

$$\sigma_{A=v}(S) = \sigma_{A=v}(F \times R) = F \times \sigma_{A=v}(R).$$

Analogously, F is a factor of $\sigma_{A \neq v}(S)$.

$\Leftarrow$. Relation F is a factor of S because

$$S = \sigma_{A=v}(S) \cup \sigma_{A \neq v}(S) = F \times E \cup F \times I = F \times (E \cup I).$$ □

algorithm prime-factorization (S)
// Input: Relation S over schema $S[U]$.
// Result: Prime factorization of S as a set Fs of its prime factors.

1. $Fs := \{\{\pi_B(S)\} \mid B \in U, |\pi_B(S)| = 1\}$; $S := S \div \prod_{F \in Fs} (F)$;
2. **if** $S = \emptyset$ **then return** Fs;
3. **choose any** $A \in \mathsf{sch}(S), v \in \pi_A(S)$ **such that** $|\sigma_{A=v}(S)| \leq |\sigma_{A \neq v}(S)|$;
4. $Q := \sigma_{A=v}(S)$; $R := \sigma_{A \neq v}(S)$;
5. **foreach** $F \in$ prime-factorization(Q) **do**
6. **if** $(R \div F) \times F = R$ **then** $Fs := Fs \cup \{F\}$;
7. **if** $\prod_{F \in Fs} (F) \neq S$ **then** $Fs := Fs \cup \{S \div \prod_{F \in Fs} (F)\}$;
8. **return** Fs;

Fig. 11. Computing the prime factorization of a relation.

Corollary 6. *A relation S is prime iff $\sigma_{A=v}(S)$ and $\sigma_{A \neq v}(S)$ are coprime.*

The algorithm prime-factorization given in Figure 11 computes the prime factorization of an input relation S as follows. It first finds the trivial prime factors with one attribute and one value (line 1). These factors represent the prime factorization of S, in case the remaining relation is empty (line 2). Otherwise, the remaining relation is disjointly partitioned in relations Q and R (line 4) using *any* selection with constant $A = v$ such that Q is smaller than R (line 3). The prime factors of Q are then probed for factors of R and in the positive case become prime factors of S (lines 5 and 6). This property is ensured by Proposition 10. The remainder of Q and R, which does not contain factors common to both Q and R, becomes a factor of S (line 7). According to Corollary 6, this factor is also prime.

Example 9. We exemplify our prime factorization algorithm using the following relation S with three prime factors.

S	A	B	C	D	E
	a_1	b_1	c_1	d_1	e_1
	a_1	b_1	c_1	d_1	e_2
	a_1	b_1	c_1	d_2	e_1
	a_1	b_1	c_1	d_2	e_2
	a_2	b_1	c_1	d_1	e_1
	a_2	b_1	c_1	d_1	e_2
	a_2	b_1	c_1	d_2	e_1
	a_2	b_1	c_1	d_2	e_2
	a_2	b_2	c_2	d_1	e_1
	a_2	b_2	c_2	d_1	e_2
	a_2	b_2	c_2	d_2	e_1
	a_2	b_2	c_2	d_2	e_2

$$\begin{array}{|ccc|} \hline \text{A} & \text{B} & \text{C} \\ \hline a_1 & b_1 & c_1 \\ a_2 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \hline \end{array} \times \begin{array}{|c|} \hline \text{D} \\ \hline d_1 \\ d_2 \\ \hline \end{array} \times \begin{array}{|c|} \hline \text{E} \\ \hline e_1 \\ e_2 \\ \hline \end{array}$$

To ease the explanation, we next consider all variables of the algorithm followed by an exponent i, to uniquely identify their values at recursion depth i.

Consider the sequence of selection parameters $(A, a_1), (D, d_1), (E, e_1)$.

The relation S^1 has no factors with one attribute. We next choose the selection parameters (A, a_1). The partition $Q^1 = \sigma_{A=a_1}(S^1)$ and $R^1 = \sigma_{A \neq a_1}(S^1))$ is shown below

Q^1	A	B	C	D	E
	a_1	b_1	c_1	d_1	e_1
	a_1	b_1	c_1	d_1	e_2
	a_1	b_1	c_1	d_2	e_1
	a_1	b_1	c_1	d_2	e_2

R^1	A	B	C	D	E
	a_2	b_1	c_1	d_1	e_1
	a_2	b_1	c_1	d_1	e_2
	a_2	b_1	c_1	d_2	e_1
	a_2	b_1	c_1	d_2	e_2
	a_2	b_2	c_2	d_1	e_1
	a_2	b_2	c_2	d_1	e_2
	a_2	b_2	c_2	d_2	e_1
	a_2	b_2	c_2	d_2	e_2

We proceed to depth two with $S^2 = Q^1$. We initially find the prime factors with one of the attributes A, B, and C. We further choose the selection parameters (D, d_1) and obtain Q^2 and R^2 as follows

Q^2	D	E
	d_1	e_1
	d_1	e_2

R^2	D	E
	d_2	e_1
	d_2	e_2

We proceed to depth three with $S^3 = Q^2$. We initially find the prime factor with the attribute D. We further choose the selection parameters (E, e_1) and obtain Q^3 and R^3 as follows

Q^3	E
	e_1

R^3	E
	e_2

We proceed to depth four with $S^4 = Q^3$. We find the only prime factor $\pi_E(Q^3) = Q^3$ with the attribute E and return the set $\{Q^3\}$.

At depth three, we check whether Q^3 is also a factor of R^3. It is not, and we infer that $Q^3 \cup R^3$ is a prime factor of Q^2 (the other prime factor $\pi_D(Q^2)$ was already detected). We thus return $\{\pi_D(Q^2), \pi_E(Q^2)\}$.

At depth two, we check the factors of Q^2 for being factors of R^2 and find that $\pi_E(Q^2)$ is also a factor of R^2, whereas $\pi_D(Q^2)$ is not. The set of prime factors of Q^1 is thus $\{\pi_E(Q^2), \pi_A(Q^1), \pi_B(Q^1), \pi_C(Q^1), \pi_D(Q^1)\}$, where $\pi_A(Q^1)$, $\pi_B(Q^1)$, and $\pi_C(Q^1)$ were already detected as factors with one attribute and one value, and $\pi_D(Q^1)\}$ is the rest of Q^1.

At depth one, we find that only $\pi_E(Q^2)$ and $\pi_D(Q^1)\}$ are also factors of R^1. Thus the prime factorization of S^1 is $\{\pi_E(Q^2), \pi_D(Q^1), \pi_{A,B,C}(S^1)\}$. The last factor is computed in line 7 by dividing S^1 to the product of the factors $\pi_E(Q^2)$ and $\pi_D(Q^1)\}$. □

Remark 4. It can be easily verified that choosing another sequence of selection parameters, e.g., (D, d_1), (E, e_1) and (A, a_1), does not change the output of the algorithm.

Because the prime factorization is unique, the choice of the attribute A and value v (line 3) can not influence it. However, choosing A and v such that $|\sigma_{A=v}(S)| \leq |\sigma_{A\neq v}(S)|$ ensures that with each recursion step the input relation to work on gets halved. This affects the worst-case complexity of our algorithm.

In general, there is no unique choice of A and v that halve the input relation. There are choices that lead to faster factorizations by minimizing the number of recursive calls and also the sizes of the intermediary relations Q. □

Theorem 10. *The algorithm of Figure 11 computes the prime factorization of any relation.*

Proof. The algorithm terminates, because (1) the input size at the recursion depth i is smaller (at least halved) than at the recursion depth $i-1$, and (2) the initial input is finite.

We first show by complete induction on the recursion depth that the algorithm is sound, i.e., it occasionally computes the prime factorization of the input relation.

Consider d the maximal recursion depth. To ease the rest of the proof, we uniquely identify the values of variables at recursion depth i ($1 \leq i \leq d$) by an exponent i.

Base Case. We show that at maximal recursion depth d the algorithm computes the prime factorization. This factorization corresponds to the case of a relation S^d with a single tuple (line 2), where each attribute induces a prime factor (line 1).

Induction Step. We know that Fs^{i+1} represents the prime factorization of $S^{i+1} = Q^i$ and show that Fs^i represents the prime factorization of S^i.

Each factor F of Q^i is first checked for being a factor of R^i (lines 5 and 6). This check uses the definition of relational division: the product of F and the division of R^i with F must give back R^i. Using Proposition 10, each factor F common to R^i and S^i is also a factor of S^i. Obviously, because F is prime in Q^i, it is also prime in S^i.

We next treat the case when the factors common to Q^i and R^i do not entirely cover S^i (line 7). Let P be the product of all factors common to Q^i and R^i, i.e., $P = \Pi Fs^i$. Then there exists Q^i_* and R^i_* such that $Q^i = Q^i_* \times P$ and $R^i = R^i_* \times P$. It follows that $S^i = Q^i \cup R^i = (Q^i_* \cup R^i_*) \times P$, thus $(Q^i_* \cup R^i_*)$ is a factor of S^i. Because Q^i_* and R^i_* are coprime (otherwise they would have a common factor), Corollary 6 ensures that their union $(Q^i_* \cup R^i_*)$ is prime.

This concludes the proof that the algorithm is sound. The completeness follows from Proposition 10, which ensures that the factors of S^i, which do not have the chosen attribute A, are necessarily factors of both Q^i and R^i at any recursion depth i. Additionally, this holds independently of the choice of the selection parameters. □

Our relational factorization is a special case of algebraic factorization of Boolean functions, as used in multilevel logic synthesis [11]. In this light, our algorithm can be used to algebraically factorize disjunctions of conjunctions of literals. A factorization is then a conjunction of factors, which are disjunctions

of conjunctions of literals. This factorization is only algebraic, because Boolean identities (e.g., $x \cdot x = x$) do not make sense in our context and thus are not considered (Note that Boolean factorization is NP-hard, see e.g., [11]).

The algorithm of Figure 11 computes prime factorizations in polynomial time and linear space, as stated by the following theorem.

Theorem 11. *The prime factorization of a relation S with arity m and size n is computable in time $O(m \cdot n \cdot \log n)$ and space $O(n)$.*

Proof. The complexity results consider the input and the temporary relations available in secondary storage.

The computations in lines 1, 3, 4, and 7 require a constant amount of scans over S. The number of prime factors of a relation is bounded in its arity. The division test in line 6 can be also implemented as

$$\pi_{sch(P)}(R) = P \text{ and } |P| \cdot |\pi_{sch(R)-sch(P)}(R)| = |R|.$$

(Here *sch* maps relations to their schemata). This requires to sort P and $\pi_{sch(P)}(R)$ and to scan R two times and P one time. The size of P is logarithmic in the size of Q, whereas Q and R have sizes linear in the size of S. The recursive call in line 5 is done on Q, whose size is at most a half of the size of S.

The recurrence relation for the time complexity is then (for sufficiently large constant a; n is the size of S and m is the arity of S)

$$\begin{aligned} T(n) &= 7n + m \cdot n \cdot \log n + T(\frac{n}{2}) \\ &\leq T'(n) = a \cdot m \cdot n \cdot \log n + T'(\frac{n}{2}) = a \cdot m \cdot \sum_{i=1}^{\lceil \log n \rceil} (\frac{n}{2^i} \cdot \log(\frac{n}{2^i})) \\ &\leq a \cdot m \cdot \sum_{i=1}^{\infty} (\frac{n}{2^i} \cdot \log(\frac{n}{2^i})) = a \cdot m \cdot n \cdot \log n = O(m \cdot n \cdot \log n). \end{aligned}$$

Each factor of S requires space logarithmic in the size of S. The sum of the sizes of the relations Q and R is the size of S. Then, the recurrence relation for the space complexity is (n is the size of S and m is the arity of S)

$$\begin{aligned} S(n) &= n + m \cdot \log n + S(\frac{n}{2}) = \sum_{i=1}^{\lceil \log n \rceil} (\frac{n}{2^i} + m \cdot \log(\frac{n}{2^i})) \\ &\leq m \cdot \sum_{i=1}^{\infty} (\frac{n}{2^i} + m \cdot \log(\frac{n}{2^i})) = n + m \cdot \log n = O(n). \end{aligned}$$

□

We can further trade the space used to explicitly store the temporary relations Q, R, and the factors for the time needed to recompute them. For this, the temporary relations computed at any recursion depth i are defined *intentionally* as queries constructed using the chosen selection parameters. This leads to a sublinear space complexity at the expense of an additional logarithmic factor for the time complexity.

Proposition 11. *The prime factorization of a relation S with arity m and size n is computable in time $O(m \cdot n \cdot \log^2 n)$ and space $O(m \cdot \log n)$.*

Proof. We can improve the space complexity result of Theorem 11 in the following way. The temporary relations computed at any recursion depth i are defined *intentionally* as queries constructed using their schema, say U^i, and the chosen selection parameters (A^i, v^i).

The relation Q^j at recursion depth $j \leq i$ is

$$Q^j = \pi_{U^j}(\sigma_{\phi_j^Q}(S)), \qquad \phi_j^Q = \bigwedge_{1 \leq l \leq j} (A^l = v^l)$$

The relation R^j is defined similarly and their factors additionally require to only store their schema. Such queries have the size bounded in the maximal recursion depth, thus in the logarithm of the input relation size. At each recursion depth, only an attribute-value pair needs to be stored. Thus the space complexity becomes (n is the size of S and m is the arity of S)

$$S(n, m) = m \cdot \log n + S(\frac{n}{2}, m-1) \leq \sum_{i=1}^{\lceil \log n \rceil} (m \cdot \log \frac{n}{2^i}) \leq \sum_{i=1}^{\infty} (m \cdot \log \frac{n}{2^i}) = m \cdot \log n.$$

The time complexity increases, however. All temporary relations need to be recomputed from the original relation S seven times at each recursion depth. Thus, in contrast to $T(n, m)$ from the proof of Theorem 11, the factor $\frac{1}{2^i}$ does not appear in the new formula of $T(n')$. The new recurrence function for $T(n')$ (for sufficiently large $a > 0$; n is the size of the initial S and m is the arity of the initial S; n' is initially n) is

$$\begin{aligned} T(n') &= 7n + m \cdot n \cdot \log n + T(\frac{n'}{2}) \\ &\leq T'(n') = a \cdot m \cdot n \cdot \log n + T'(\frac{n'}{2}) = \sum_{i=1}^{\lceil \log n \rceil} (a \cdot m \cdot n \cdot \log n) \\ &= a \cdot m \cdot n \cdot \log^2 n = O(m \cdot n \cdot \log^2 n). \end{aligned}$$

□

Remark 5. An important property of our algorithm is that it is polynomial in both the arity and the size of the input relation S. If the arity is considered constant, then a trivial prime factorization algorithm (yet exponential in the arity of S) can be devised as follows: First compute the powerset $PS(U)$ over the set U of attributes of S. Then, test for each set $U' \in PS(U)$ whether $\pi_{U'}(S) \times \pi_{U-U'}(S) = S$ holds. In the positive case, a factorization is found with factors $\pi_{U'}(S)$ and $\pi_{U-U'}(S)$, and the same procedure is now applied to these factors until all prime factors are found. Note that this algorithm requires time exponential in the arity of the input relation (due to the powerset construction). Additionally, if the arity of the input relation is constant, then the question whether a relation S is prime (or factorizable) becomes FO-expressible (also supported by the space complexity given in Proposition 11). □

6.3 Optimization Flavors

The gWSD optimization discussed here is a facet of the more general problem of finding minimal representations for a given g-tabset or world-set. To find a minimal representation for a given g-tabset $\mathbf{A}$, one has to take into account all possible inlinings for the g-tables of $\mathbf{A}$ in g-tabset tables. Recall from Section 3 that we consider a fixed arbitrary inlining order of the tuples of the g-tables in $\mathbf{A}$. Such an order is supported by common *identifiers* of tuples from different worlds, as maintained in virtually all representation systems [19, 3, 17, 8] and *exploited* in practitioner's representation systems such as [8, 4]. We note that when no correspondence between tuples from different worlds has to be preserved, smaller representations of the same world-set may be possible.

Acknowledgments. The authors were supported in part by DFG project grant KO 3491/1-1. The first author was supported by the International Max Planck Research School for Computer Science, Saarbrücken, Germany.

References

1. S. Abiteboul and O. M. Duschka. "Complexity of Answering Queries Using Materialized Views". In *Proc. PODS*, pages 254–263, 1998.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.
4. P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *Proc. ICDE*, 2006.
5. L. Antova, C. Koch, and D. Olteanu. 10^{10^6} worlds and beyond: Efficient representation and processing of incomplete information. In *Proc. ICDE*, 2007.
6. L. Antova, C. Koch, and D. Olteanu. World-set decompositions: Expressiveness and efficient algorithms. In *Proc. ICDT*, pages 194–208, 2007.
7. M. Arenas, L. E. Bertossi, and J. Chomicki. "Answer sets for consistent query answering in inconsistent databases". *TPLP*, **3**(4–5):393–424, 2003.
8. O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proc. VLDB*, 2006.
9. L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. "Complexity and Approximation of Fixing Numerical Attributes in Databases Under Integrity Constraints". In *Proc. DBPL*, pages 262–278, 2005.
10. P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. "A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification". In *Proc. SIGMOD*, June 2005.
11. R. K. Brayton. Factoring logic functions. *IBM J. Res. Develop.*, 31(2), 1987.
12. D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. "Logical Foundations of Peer-To-Peer Data Integration". In *PODS 2004*, pages 241–251, 2004.
13. J. Chomicki, J. Marcinkowski, and S. Staworko. "Computing consistent query answers using conflict hypergraphs". In *Proc. CIKM*, pages 417–426, 2004.
14. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *Proc. VLDB*, pages 864–875, 2004.
15. M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

16. G. Grahne. Dependency satisfaction in databases with incomplete information. In *Proc. VLDB*, pages 37–45, 1984.
17. G. Grahne. *The Problem of Incomplete Information in Relational Databases*. Number 554 in LNCS. Springer-Verlag, 1991.
18. T. J. Green and V. Tannen. "Models for Incomplete and Probabilistic Information". In *International Workshop on Incompleteness and Inconsistency in Databases (IIDB)*, 2006.
19. T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of ACM*, 31:761–791, 1984.